or a representational state transfer protocol (REST) that enables the client application **205** to access and/or operate on resources associated with the service. The API can enable the client application **205** to transmit data to and/or receive data from the service. API calls can also cause the service to perform various operations and/or call additional services using additional API calls.

[0044] In other embodiments, the client application **205** is a web-based application displayed within a browser of a client device **120**. The client application **205** can include a series of resources (e.g., hypertext markup language (HTML) documents, images, scripts, etc.) requested from a server associated with a website. The browser receives the resources and interprets the resources to display a representation of the website on a display of the client device **120**. The client application **205** is therefore platform-independent in that the client application **205** can be displayed on a plurality of different client devices **120** running different operating systems.

[0045] In some embodiments, the client application **205** interfaces with the one or more services through an API gateway **210**. The API gateway **210** is implemented by a server device **110** and redirects API calls received from client devices **120** to the various services in the client-server architecture **200**. In other embodiments, the API gateway **210** is omitted or selectively bypassed and the client devices **120** transmit API calls to the services directly.

[0046] In some embodiments, the client application **205** enables an instructor to create hand-outs for a class. A hand-out refers to a file or data structure that includes information related to an assignment intended to be published to students within the class. The hand-out can include a number of fields including: a hand-out identifier, a title of the assignment, a body of the assignment that includes text-based instructions for the students on how to complete the assignment, a due date for the assignment, and a list of attachments associated with the assignment. The hand-outs can be created and stored locally on a particular client device **120**. The attachments can include files (e.g., documents, images, videos, etc.), placeholders for a file the student is to turn in, and activities the student is to complete as part of the assignment. The activities are performed using third-party applications that implement a portion of a class kit software framework. Examples of activities can include, but are not limited to, reading a chapter of a digital book or textbook, taking a quiz or answering a set of problems, tracking time spent performing a task such as playing an interactive game or performing a digital experiment, and so forth.

[0047] In some embodiments, the client-server architecture **200** includes a hand-out service **220**. The hand-out service **220** is configured to manage hand-outs. In some embodiments, the hand-out service **220** is configured to sync hand-outs created on one client device **120** with another client device **120** to enable an instructor to work on multiple devices. In some embodiments, the hand-out service **220** is configured to enable the hand-outs to be published to a list of students. Publishing a hand-out refers to making the information in the hand-out available to be viewed by the students on a separate client device **120** as well as handling various back-end operations related to the attachments for the hand-outs.

[0048] In some embodiments, the client-server architecture **200** can also include a hand-in service **230**. A hand-in refers to a placeholder for a file or data structure, which indicates that a student is instructed to create a file or data structure to turn in in order to complete the assignment. A student can create the file or data structure and submit the file or data structure to the hand-in service **230** to satisfy the requirement for completing the assignment.

[0049] In some embodiments, the client-server architecture **200** can also include a school management service **240** configured to manage administrative information for a school district related to the structure of classes. For example, the school management service **240** can maintain records that indicate which instructors are assigned to each of a plurality of classes. The records can also indicate which students are enrolled in each of the plurality of classes. Each instructor or student can be assigned an instructor identifier or a student identifier, respectively. Each class created by the school district can be associated with a class identifier. A separate record can then be created for each class identifier that lists a roster of student identifiers for students enrolled in the class. In some embodiments, a relational database associates instructor identifiers and/or student identifiers with class identifiers in one or more tables. The relational database can be queried using Structured Query Language (SQL) or some other type of query language to return information that identifies the structure of various classes.

[0050] In some embodiments, the school management service **240** includes an administrative interface that enables an administrator for a school district to create classes and specify the students enrolled in the class. The administrative interface can be, e.g., a web-based interface requiring the administrator to provide credentials in order to change the rosters for each class. In other embodiments, the school management service **240** includes an interface to download data from a separate and distinct school information system that the school district maintains separately from the client-server architecture **200**. The classes and rosters can be automatically downloaded from the school information system.

[0051] In some embodiments, the client-server architecture **200** can also include a progress pipeline **250**. The progress pipeline **250** provides a secure architecture for managing information related to progress tracking as students complete the assignments described in the hand-outs published by the instructors. The client devices **120** of the students can include a background process (i.e., a daemon) configured to monitor activities related to one or more third-party applications installed on the client devices **120**. The daemon tracks progress made by the students in completing the assignments specified in one or more published hand-outs and transmits information related to the progress of each student to the progress pipeline **250**. The progress pipeline **250** aggregates and stores the progress information to enable the instructor to view comprehensive reports for one or more students in a class.

[0052] In some embodiments, the progress pipeline **250** is implemented as a number of separate services executing on different server devices **110** and structured to process progress information in a dataflow in a pipelined manner. In other embodiments, the progress pipeline **250** can be implemented with a number of services executing on a single server device **110**.

[0053] In some embodiments, the client-server architecture **200** can also include an identity service **260**. The identity service **260** enables data related to particular client devices **120** to be associated with particular people (e.g.,